ISBN: 978-972-8939-72-4 © 2012 IADIS

ADVANCED VIDEOCONFERENCING SERVICES BASED ON WEBRTC

Pedro Rodríguez, Javier Cerviño, Irena Trajkovska and Joaquín Salvachúa
Universidad Politécnica de Madrid
Ciudad Universitaria, Avda. Complutense s/n
28040, Madrid, Spain

ABSTRACT

Lately, videoconference applications have experienced an evolution towards the World Wide Web. New technologies have given browsers real-time communications capabilities. In this context, WebRTC aims to provide this functionality by following and defining standards. Being a new effort, WebRTC still lacks advanced videoconferencing services such as session recording, media mixing and adjusting to varying network conditions. This paper analyzes these challenges and proposes an architecture based on a traditional communications entity, the Multipoint Control Unit or MCU as a solution.

KEYWORDS

Videoconference, WebRTC, Multimedia, MCU.

1. INTRODUCTION

Throughout the last decade we have witnessed a great evolution of the World Wide Web. Web pages have transitioned from static information to full fledged applications with great interactivity and functionality. Videoconference has also been affected by this trend. Multimedia communication applications started to pop up in our web browsers enabled by proprietary plug-ins such as Adobe Flash. While these products in many cases provided complete videoconferencing and collaborative experiences, they frequently relied on proprietary solutions and protocols imposed by the developer of the browser plug-in instead of well-known standards.

Lately, effort has been put to provide web browsers and pages with more functionality and interactivity without the need of relying on plug-ins. HTML5 aims to provide these advanced features to web pages by updating the standard markup language and adding JavaScript APIs. In this context, WebRTC is being defined and developed to offer real-time peer-to-peer communications to the web taking advantage of HTML5 and existent real-time protocols and codecs instead of defining new ones. WebRTC is a joint effort by the WebRTC working group of the World Wide Web Consortium (W3C) and the rtcweb group from the Internet Engineering Task Force (IETF) where the first provide the HTML and JavaScript API and the latter defines the protocols and codecs to be used in the communication.

As of now, the first implementations of WebRTC are being developed. However, the definition is not finished and there is room for innovation when it comes to advanced communication services.

In this paper we provide an overview of videoconferencing via WebRTC as opposed to previous solutions for real-time communications within web browsers. Secondly, we enumerate the challenges that appear when using this new technology. Then, we propose a solution for these challenges by re-introducing a well-known entity in the communications world, the Multipoint Control Unit or MCU. Finally, we present an architecture for this MCU and show our first steps towards implementing it.

2. WEB VIDEO CONFERENCE CONTEXT

Rich Internet Applications (RIAs)[2] are seen as the evolution of static web pages. These applications provide animations of text, drawings, handling of user input, drag and drop functionality, and bi-directional streaming of audio and video among other characteristics. Adobe Flash, Oracle JavaFX and Microsoft Silverlight are currently the most common platforms, with more than 50% market penetration. As opposed to this proprietary solutions, the W3C is working on defining the next version of the HTML markup language, HTML5 which provides more desktop-like features to web pages in a standard way.

WebRTC is the joint effort of both the W3C and IETF to provide web browsers real-time peer-to-peer communication. These groups aim to specify a standard set of protocols (IETF) and JavaScript APIs (W3C) that offer real-time communication capabilities for web developers to use in their applications. As opposed to previous alternatives, WebRTC is being defined to use well-known and tested standards instead of proprietary solutions.

The most relevant decisions that have been taken at the time of writing this paper are:

- ICE [9] with STUN and TURN will be used for Network Address Translation (NAT) Traversal.
- RTP [11] and its secure variant SRTP [3] are used for data framing.
- Media negotiation will be done via SDP [5]
- No signaling protocol (such as SIP [8]) will be recommended in order to give developers more flexibility for innovation in web applications.

While the process of definition is not over, implementations are starting to be available and videoconferencing is seen as the first milestone.

In the following sections we will overview the details of WebRTC when it comes to videoconferencing and the challenges that come with it as well as propose a solution for them.

3. CHALLENGES

The implementation of a known technology in a new environment brings a host of challenges. Previous experiences ([4] and [7]) provide us with enough background and knowledge to foresee challenges that are going to be solved before video conferencing through WebRTC becomes widely used in advanced applications that go beyond one on one conversation. These challenges rise from the web environment, where, ideally, different kinds of devices have to interact in an efficient way and from providing higher level services, needed in complex scenarios like education:

Heterogeneous access networks and devices: The diversity of devices accessing the Web has increased considerably over the last years. Where once the traditional personal computer was the only significant access point to the WWW there are now a number of devices with different capabilities and, in many cases, access networks. Such is the case of smartphones, which still have limited processing power compared to desktop computers and usually access the web via cellular networks.

Screen size: Smaller screens cannot show the same amount of information as big ones. In video conferencing that means, for instance, sending a really high quality video to a small screen device is inefficient, as users cannot tell the difference. A more complicated scenario is multi conferencing where the screen size limits the amount of participants that can be shown at the same time.

CPU: Video conferencing requires processing power to decode, encode and distribute video and audio in real time. This CPU stress depends on several factors such as the codecs used, the quality of both audio and video and video size. In mobile systems CPU power not only imposes a limitation in the things that can be done but also, stressing the device too much can lead to an perceivable decrease in battery life.

Bandwidth availability and latency: As we mentioned above, the variety of devices goes in hand with an array of different access networks. For instance, we have typical wired Ethernet access from desktop computers and 3G networks for mobile devices. These differences have to be taken into account if we want to optimize the communication. For instance, 3G connections can vary the available bandwidth without any notice. That will result in an interruption in the conference if the system is not able to react accordingly.

Session recording: In some scenarios such as enterprise meetings and e-learning, recording the entire information generated in a videoconference session is an essential feature. While the current WebRTC

definition is explicit when it comes to self-video recording it does not provide a mechanism to gather the streams from all the participants and store them.

Gateways: The interoperability of one communication platform with another is always a challenge. For instance, communication between SIP Phones and traditional Public Switched Telephone Networks (PSTN) networks is achieved via gateways that translate signaling and media streams.

4. MCU

To address the challenges listed above, we are working on implementing a Multipoint Control Unit (MCU) able to redirect, adapt and translate media streams.

At the time of writing this paper, the rtcweb IETF working group has already defined most aspects of a videoconference at a protocol level as well as the media negotiation. Accordingly, implementing a MCU should be a safe route as it operates at transport and below levels leaving flexibility on the application level rather than depending on the evolution of the definition of the JavaScript API by the W3C.

We propose the development of a new MCU that abides to the standards and is able to communicate in the WebRTC world as the enabler to provide the advanced services hinted in the previous section.

4.1 MCU: Multipoint Control Unit

The MCU is an integral component of multipoint videoconference systems since their inception. It is a central device that interconnects the multiple participants of a videoconference as seen in [12]. It is a requirement when clients can only establish a single point connection.

In the context of this paper, the MCU is a piece of software that, by abiding to the definitions of the IETF, can seamlessly communicate with WebRTC peers while being able to redirect and transcode media streams at will. The MCU can be located anywhere in the network and it does not represent a participant in the conference, but a special entity that performs operations in order to provide advanced communication services.

Even if WebRTC peers could establish more than one connection forming an overlay network, the role of a MCU still has sense and it is vital to overcome the challenges mentioned in section 3.

In the simplest scenario, a MCU only has to redirect media flows among the participants so they can receive data from others acting as the center of a star topology.

However, as all data is going to go through that MCU, several operations can be providing advances services for videoconferencing.

Media Transcoding: The use of a more advanced MCU able to mix and transcode media streams can pave the way to solving the heterogeneity of devices and access networks. By transcoding streams into different bit-rates and sizes, the communication can be adapted to diverse network conditions and screen sizes optimizing the use of the network and CPU usage in the participants at the expense of the MCU. This is also useful in a gateway scenario where media streams have to be translated.

Media Composing: By generation a single video and audio stream from the available inputs, the MCU can reduce the amount of CPU overhead and control needed to participate in a multi-participant videoconference when needed.

Media Recording: The MCU is receiving all the streams present in the conversation and, as stated in the previous point, is able to generate a composed stream by combining them. If a recording of the session is required, the MCU can store that stream for future reproductions.

These capabilities allow us to provide advanced services often requested in multiconferencing and collaborative applications by controlling all the information available in a session.

4.2 WebRTC-MCU Architecture

In this subsection we will provide an overview of the architecture of the proposed MCU, capable of interacting with current WebRTC implementations.

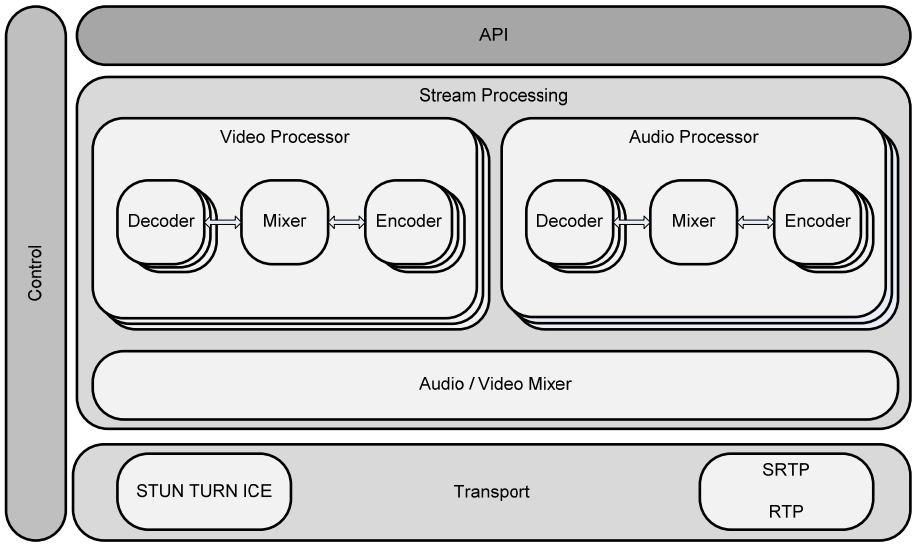

Figure 1. WebRTC-MCU Architecture

Conceptually, the MCU has four big components: API, stream processing, transport and control:

API: It exports the functionality provided by the rest of the components. It has to provide applications functions to specify the streams that have to be received, the operations to be performed on them, and the output streams. This layer also includes the abstract signaling present in WebRTC in order to be able to establish media communications with the participants.

Transport: All transport related functions are implemented in this module. This is the most important part when it comes to compatibility with WebRTC. It implements ICE with STUN and TURN, RTP and SRTP.

Stream Processing: This component comprises all the media processing to be performed by the MCU. It is capable of decoding, processing, mixing and encoding. It is capable of processing one or more streams and producing one or more outputs that will be sent via the transport component. As shown in Figure 1, this is structured by dividing the processing into audio and video processing components. Each of those contains one or more decoders, a mixer and one or more encoders. Finally, an audio/video mixer exists in order to be able to produce streams that contain both audio and video.

Control: This component is in charge of executing the global vision of the function specified via the API. It has total control over the Stream Processing and Transport layers. It connects the streams received via transport with the properly configured steam processors.

4.3 WebRTC-MCU Implementation

We have developed a prototype of the implementation of the MCU proposed above. In this first step, the focus is in the transport component, the one that has to follow the standards more closely and the most critical part when it comes to being compatible with the current WebRTC implementation.

We are focusing on using well known, tested and documented open-source C libraries such as libnice[1] and libsrtp[2] for ICE and SRTP respectively. However, working with WebRTC at this early stage has its challenges. The constant evolution of both the standard and the implementation produces a changing environment as well as a not very strict following of the standards. The main difficulty at this point is to adapt to these changes and refine the communication so the MCU can look through these adjustments in the otherwise well-known standards.

At the time of writing this paper, the prototype allows for retransmitting a given webrtc stream to other participants.

[1] http://nice.freedesktop.org
[2] http://srtp.sourceforge.net/

5. CONCLUSION AND FUTURE WORK

In this paper we propose a centralized video conferencing architecture based on a MCU for WebRTC environments. This architecture supports the participation of multiple participants in a video conferencing session using a single connection. We discussed how such architecture provides solutions to specific scenarios with requisites like session recording, stream processing and composition for bandwidth and screen adaptation. The paper also shows current challenges we have found during the implementation of this MCU and what features need to be supported in WebRTC libraries. Features like the participation of multiple users, the recording of sessions, transcoding, etc.

We describe which components are needed for implementing WebRTC-capable applications and what differences exist between standard and implementations at the time of writing this paper. And we also explain our first implementation of a MCU, which takes into account these requisites. With this implementation we wanted to learn and show how WebRTC works, and it serves us and other developers as a good starting point for implementing additional features and testing centralized web video conferencing architectures.

Despite of having a first MCU implementation, we pretend to extend its functionality by developing new features: video and audio stream processing for supporting different kind of devices (mobile phones, tablets, and desktops), gateways for communicating WebRTC clients such as SIP, XMPP [10] or even H.323 [6] clients, session recording, streaming.

ACKNOWLEDGEMENTS

This research project was made possible thanks to funding from the SAAN project (TIN2010-19138)

REFERENCES

[1] Adobe Systems Inc., 2009. Real-Time Messaging Protocol (RTMP). *Adobe Specification.*

[2] J. Allaire, 2002. Macromedia Flash MX-A next-generation rich client. *Macromedia White Papers.*

[3] M. Baugheret al., 2004. The Secure Real-time Transport Protocol (SRTP). *RFC IETF.*

[4] J. Cerviño et al, 2011. Videoconference Capacity Leasing on Hybrid Clouds. *IEEE 4th International Conference on Cloud Computing*, pages 340-347, Washington D.C., USA.

[5] M. Handley et al., 2006. RFC 4566: SDP: Session Description Protocol. *IETF RFC.*

[6] ITU-T, 2009. Rec. H.323, Packet-based multimedia communication systems. *H SERIES: AUDIOVISUAL AND MULTIMEDIA SYSTEMS.*

[7] P. Rodriguez et al., 2009. VaaS: Videoconference as a service. *CollaborateCom 2009. 5th International Conference on,* pages.1-11, Washington D.C., USA.

[8] J. Rosenberg et al, 2002. RFC 3261: SIP: Session Initiation Protocol. *IETF RFC.*

[9] J. Rosenberg, 2010. RFC 5245: Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. *IETF RFC.*

[10] P. Saint-Andre, 2011. RFC: 6120 Extensible Messaging and Presence Protocol (XMPP): Core. *IETF RFC.*

[11] H. Schulzrinne, et al, 2003. RFC 3550: RTP: A Transport Protocol for Real-Time Applications. *IETF RFC.*

[12] M.H. Willebeek-LeMair, 1994. On multipoint control units for videoconferencing. *Local Computer Networks, 1994. Proceedings., 19th Conference on*, pages 356 – 364, Minneapolis, MN , USA.